Annual Progress Report

NASA Grant NAG 2-893

Formal Support for High Assurance Systems

Principal Investigator: Fred B. Schneider

Department of Computer Science
Cornell University
Ithaca, New York 14853

January 23, 1996

1. Introduction

Progress was made over the past year in our investigations regarding programming logics, paradigms, and protocols for implementing fault-tolerant and distributed systems. Some of the work in formal logics was foundational, with an eye towards developing a predicate logic that is convenient and flexible; other of the work concerned the development proof rules for reasoning about causally-ordered message delivery, a communications service that underlies many of today's distributed-systems toolkits. Our paradigms and protocols work has led to a new and optimal algorithm for property detection and an implementation of hypervisor-based fault-tolerance.

2. Equational Propositional Logic

David Gries and I have developed an equational predicate logic, called E. We are primarily interested in the use of the logic; the equational style makes it possible to develop and present calculations in a rigorous manner, without complexity and detail overwhelming (in contrast to other proof styles). Logic E is the basis for a sophomore-level discrete mathematics textbook we wrote. That text is now being used at 40 or so departments, suggesting that the text's authors are not the only ones who find the approach attractive. The text's theme is that logic can be a tool rather than simply an object of study.

Two open issues regarding E were pursued over the past year. First, we proved that the equational propositional logic underlying E is sound and complete [1]. Second, in [7], we explored the treatment of undefined expressions and partial functions. Partial functions are

ubiquitous in programming—some basic mathematical operations are partial (e.g. division), some basic programming operations are partial (e.g. array subscripting), and many functions that arise through recursive definitions are partial. Therefore, a logic for reasoning about programs must handle partial functions. The approach we embrace is to regard all variables, terms, and predicates as being total, but in some cases underspecified.

3. Causally-Ordered Message Delivery

Causally-ordered delivery can be understood as a generalization of FIFO ordering. With both, a message is delivered only after all messages on which it may depend. With FIFO ordering, this guarantee applies only to messages having the same sender; with causal ordering, this guarantee applies to messages sent by any process. In [3], we give a Hoare-style proof system for causally-ordered delivery and use the logic to verify an asynchronous variant of the distributed termination detection algorithm developed by Dijkstra, Feijen, and van Gasteren.

Our proof system is similar in style to the satisfaction-based logics that have been proposed for asynchronous and synchronous message-passing. Our logic is interesting because reasoning about message-passing primitives for causally-ordered delivery involves a global property, the system-wide causality relation, which defines what messages are deliverable. We thus demonstrate that substantially new methods are not required when message-delivery semantics depends on global information. Others had suggested that new methods would be required for this case.

4. Distributed Property Detection

Execution of a distributed system can be modeled as a sequence of events in their order of occurrence. Such a sequence uniquely determines the global states through which the system has passed. Unfortunately, in an asynchronous distributed system, no process can determine the order in which events on different processors actually occur. Therefore, no process can determine the sequence of global states through which the system passed. This leads to an obvious difficulty for detecting whether a global state predicate held.

In [5], we present a new algorithm for detecting whether a particular execution of an asynchronous distributed system satisfies **poss** Φ (read "possibly Φ") meaning the system could have passed through a global state satisfying Φ. The algorithms allows Φ to be any global state predicate and is optimal when Φ has a certain structure. We also give an off-line algorithm for detecting **poss** Φ, based on Strassen's scheme for fast matrix multiplication.

5. Hypervisor-based Fault-tolerance

Software is now running that implements a new approach for making a computer processor fault-tolerant [9]. We built and augmented the functionality of a virtual-machine monitor (hypervisor). By augmenting a hypervisor, fault-tolerance was achieved without modifying the hardware, the operating system, or any application code.

Our protocols control two replicas of a processor and ensure that the sequence of instructions executed by two virtual machines running on different physical processors are identical. The protocols also coordinate I/O issued by these virtual machines so that the environment—disks, networks, etc—is unaware of the replicated processors and instruction streams.

Use of a hypervisor to implement replica coordination is attractive for pragmatic reasons. When replica coordination is implemented in a hypervisor, the functionality instantly becomes available to all hardware realizations of the given instruction-set architecture, including realizations that did not exist when the hypervisor was written. Second, when replica coordination is implemented in a hypervisor, a single implementation suffices for every operating system that executes on that instruction-set architecture. Finally, by implementing replica coordination in a hypervisor, applications programmers are freed from this task.

Our prototype implements a fault-tolerant Hewlett Packard PA-RISC processor running the HP-UX operating system. We support a single SCSI disk device and an X-Windows console. Performance experiments and modeling indicate that only a modest performance degradation results from our replication management protocols.

6. Interactions with ARPA and NASA

(1) NASA/Langley currently funds an effort at ORA Corp. to help Union Switch and Signal Company (USS) employ formal methods in developing next-generation railroad switching equipment. I continue to help out. One task is to verify that a new train-control scheme, so-called "moving block" track occupancy rules, cannot lead to collisions. A second task is to study the correctness of the USS software to translate between a description of a railroad track system and the control programs that set switches and signals.

(2) One of ARPA's four ISAT studies this past year dealt with Defensive Information Warfare. I was a member of the study team. The concern is how we might protect the national infrastructure—which is increasingly dependent on distributed computing—from accidental failures and malicious intrusions. The study dealt with technology for implementing fault-tolerance and security.

7. Reports and Publications

(1) Equational propositional logic. *Information Processing Letters* 53,3 (February 1995), 145-152. With David Gries.

(2) Operating system support for mobile agents. *Proc. Fifth Workshop on Hot Topics in Operating Systems HOTOS-V* (Orcas Island, Washington, May 1995), 42-45. With Dag Johansen and Robbert van Renesse.

(3) Verifying programs that use causally-ordered message-passing. *Science of Computer Programming* 24,2 (1995), 105-128. With Scott Stoller.

(4) A new approach to discrete teaching mathematics. *Primus* V,2 (June 1995), 113-138. With David Gries.

(5) Faster possibility detection by combining two approaches. *Proc. 9th International Workshop, WDAG '95*, (Le Mont-Saint-Michel, France, Sept. 1995) Lecture Notes in Computer Science, Volume 972, Springer-Verlag, New York, 1995, 318-332. With Scott Stoller.

(6) Teaching math more effectively, through the design of calculational proofs. *The Mathematical Monthly* (October 1995), 691-697. With David Gries.

(7) Avoiding the undefined by underspecification. *Computer Science Today Recent Trends and Developments* (Jan van Leeuwen, ed). Lecture Notes in Computer Science, Vol. 1000, Spring-Verlag, 1995, 366-373. With David Gries

(8) Avoiding AAS Mistakes. *Proceedings of the Air Traffic Management Workshop* (eds. L. Tobais, M. Tashker, A. Boyle), NASA Conference Publication 10151, NASA Ames Research Center, Feb, 1995, 133-149.

(9) Hypervisor-based Fault Tolerance. *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, Operating Systems Review 29, 5 (Copper Mountain Resort, Colorado, Dec. 1995), 1-11. With T. Bressoud.

NASA Grant NAG 2-893

"Formal Support for High Assurance Systems"

Principal Investigator: Fred B. Schneider

Research Plans: 1996

o Develop methods using standard data-flow analysis techniques to determine whether the output of a system changes in response to certain classes of failures. Implement a prototype tool based on those analysis techniques and having a graphical front-end for use with systems constructed from TACOMA agents.

o Develop protocols for implementing active replication in a system of TACOMA agents. Define language constructs so a TACOMA programmer can exploit active replication without coding it directly.

o Axiomatize Dijkstra's "everywhere" operator for equational predicate logic E. Prove completeness of the axiomatization and relate it to standard modal logics S4 and S5.

o Assist in creating TLA specifications for the ARPA-funded HORUS system. Determine methods for programmers to view these specifications using a hypertext representation. Create ways that the specifications can be employed as part of an intelligent "link editor" to allow synthesis of custom HORUS stacks.